BAB II
TINJAUAN PUSTAKA

2.1 Teori Judul

2.1.1 Pengertian Sistem

Menurut Kristanto (2018:1), "suatu sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu".

Sedangkan menurut Anggraeni dan Irviani (2017:1), "sistem adalah kumpulan orang yang saling bekerja sama dengan ketentuan-ketentuan aturan yang sistematis dan terstruktur untuk membentuk satu kesatuan yang melaksanakan suatu fungsi untuk mencapai tujuan".

Berdasarkan pengertian diatas dapat disimpulkan bahwa sistem adalah jaringan kerja yang saling berhubungan dengan ketentuan-ketentuan aturan yang sistematis dan terstruktur untuk mencapai tujuan tertentu.

2.1.2 Pengertian Kelayakan

Menurut Sudarmo dkk (2018:8), "Kelayakan merupakan hal-hal yang dapat diterima, dapat dicapai, atau dapat dikerjakan, dapat diperoleh, atau yang dapat memberikan kepuasan dari sebuah objek pengamatan kepada pengamat".

Sedangkan menurut Wicaksono (2017:174), "Kelayakan merupakan sebuah gabungan dari berbagai disiplin ilmu, baik dari akuntansi maupun manajemen, dalam lingkup analisa kebutuhan perangkat lunak, lebih fokus kepada kelayakan finansial".

Berdasarkan pengertian diatas dapat disimpulkan bahwa kelayakan adalah hal-hal yang dapat diterima, dapat dicapai, atau dapat dikerjakan merupakan gabungan dari berbagai disiplin ilmu, baik dari akuntansi maupun manajemen.

2.1.3 Pengertian Kenaikan Jabatan

Menurut Adnan (2016:2), "Kenaikan (assessment) jabatan adalah program penilaian kompetensi karyawan guna di promosikan ke level diatasnya. Kenaikan jabatan menggunakan metode yang beragam tergantung kebijakan dari perusahaan".

Sedangkan menurut Maryoto (2019:12), "Kenaikan pangkat atau kenaikan jabatan adalah penghargaan yang diberikan kepada pegawai negeri sipil atas prestasi kerja dan pengabdiannya".

Menurut Soekarso dan Putong (2015:6), "Kenaikan jabatan adalah prestasi sehingga harus di kejar dengan cara apapun".

Dari ketiga definisi diatas dapat disimpulkan bahwa kenaikan jabatan adalah program penilaian yang diberikan kepada pegawai atau karyawan atas prestasi kerja dan pengabdiannya sehingga harus dikejar.

2.1.4 Metode *Simple Multi Attribute Rating Technique* (SMART)

Menurut Nofriansyah dan Defit (2017:27), mengatakan bahwa SMART (*Simple Multi Attribut Rating Technique*) merupakan metode dalam pengambilan keputusan multiatribut yang dikembangkan oleh Edward pada tahun 1997. Teknik pembuatan keputusan multiatribut ini digunakan untuk mendukung pembuat keputusan dalam memilih antara beberapa alternatif. Setiap pembuat keputusan harus memilih sebuah alternatif yang sesuai dengan tujuan yang telah dirumuskan. Setiap alternatif terdiri dari sekumpulan atribut dan setiap atribut mempunyai nilai atau bobot dan setelah itu di rata-rata dengan skala tertentu.

Menurut Diana (2018:73), metode SMART (Simple Multi Attribute Rating Technique) merupakan metode pengambilan keputusan yang dikembangkan oleh Edward pada tahun 1977. SMART merupakan teknik pengambilan keputusan multikriteria ini didasarkan pada teori bahwa setiap alternative terdiri dari sejumlah kriteria memiliki bobot yan menggambarkan seberapa penting ia dibandingkan dengan kriteria lain. Pembobotan ini digunakan untuk menilai setiap alternative agar diperoleh alternatif terbaik.

Dari kedua definisi diatas dapat disimpulkan bahwa metode SMART merupakan teknik dalam pengambilan keputusan yang digunakan untuk mendukung pembuat keputusan dalam memilih antara beberapa alternative, terdiri dari sekumpulan atribut dan setiap atribut memiliki nilai atau bobot

Model yang digunakan dalam SMART (*Simple Multi Attribut Rating Technique*) (Nofriansyah dan Defit, 2017:27) yaitu:

$$U(a_i) = \sum w_{J=1}^{m}\ Wj\ Ui\ (ai)$$

Keterangan:

1. W_j = Nilai Pembobotan Kriteria ke- j dan K- kriteria
2. $U(a_i)$ = nilai Utility kriteria ke-I untuk kriteria ke-i

   Dimana i= 1,2,…….m

Adapun algoritma penyelesaian dari Metode SMART (*Simple Multi Attribute Rating Technique*) yaitu sebagai berikut:

1. Langkah 1 : Menentukan Jumlah Kriteria dari Keputusan yang akan di ambil
2. Langkah 2 : Sistem secara default memberikan nilai 0-100 berdasarkan prioritas dengan melakukan normalisasi ($W_j / \sum W_j$)
3. Langkah 3 : Memberikan nilai kriteria untuk setiap alternatif
4. Langkah 4 : Menghitung nilai Utility untuk setiap kriteria masing- masing

$$u_i(a_i) = 100\,\frac{C_{max} - C_{out\ i}}{C_{max} - C_{min}}\ \%.....$$

Dimana:

1) $u_i(a_i)$ adalah nilai utiliti kriteria ke-1 untuk kriteria ke–I,
2) C_{max} adalah nilai kriteria maksimal
3) C_{min} adalah nilai kriteria minimal
4) $C_{out\ i}$ adalah nilai kriteria ke- i.

5. Langkah 5 : Menghitung nilai akhir dan melakukan Perangkingan.

2.2 Teori Program

2.2.1 Basis Data (*Database*)

Menurut Pamungkas (2017:2) “Basis data merupakan suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media, yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, dan dengan software untuk melakukan manipulasi untuk kegunaan tertentu”.

Menurut Jubilee (2017:1) mengatakan bahwa basis data adalah suatu aplikasi yang menyimpan sekumpulan data. Setiap basis data mempunyai perintah tertentu untuk membuat, mengakses, mengatur, mencari, dan menyalin data yang ada di dalamnya.

Sedangkan menurut Firly (2019:110), “*Database* adalah sekumpulan informasi yang diorganisasikan sehingga mudah diakses, dikelola dan diperbaharui”.

Dari ketiga definisi di atas dapat disimpulkan bahwa basis data adalah sekumpulan data yang disimpan secara bersama-sama dan setiap basis data mempunyai perintah tertentu untuk membuat, mengakses, mengatur, mencari, dan menyalin data yang ada di dalamnya.

2.2.2 Pengertian *MySQL*

Menurut Kadir (2018:170), “MySQL (baca: “Mai-es-kyu-el”) merupakan sistem manajemen database terkenal yang sekarang dimiliki oleh Oracle dan salah satu produknya bernama MySQL Community Server bersifat *Open Source*”.

Menurut Mundzir (2018:217) “MySQL adalah sistem manajemen *database* SQL yang sifatnya *open source* (terbuka) dan paling banyak digunakan saat ini. Sistem *database* MySQL mampu mendukung beberapa fitur seperti *multithreaded, multi-user,* dan SQL *database management system* (DBMS).

Menurut Siahaan dan Sianipar (2020:28), “MySQL merupakan sebuah sistem database relasional, sehingga dapat mengelompokkan informasi ke dalam tabel-tabel, atau grup-grup informasi yang berkaitan”.

Jadi dapat disimpulkan, MySQL merupakan sistem atau media yang berguna untuk mengelola database, MySQL adalah sistem manajemen *database*

yang sifatnya *open source* (terbuka) dan paling banyak digunakan dan memiliki database yang relasional sehingga dapat mengelompokkan informasi ke dalam tabel atau grup yang berkaitan.

2.2.3 *Hypertext Preprocessor* (PHP)

Menurut Yudhanto dan Prsetyo (2018:7), "PHP adalah bahasa pemrograman script server side yang sengaja dirancang lebih cenderung untuk membuat dan mengembangkan web. Bahasa pemrograman ini memang dirancang untuk para pengembang web agar dapat menciptakan suatu halaman web yang bersifat dinamis".

Sedangkan menurut Enterprise (2018:1), "PHP merupakan bahasa pemrograman yang digunakan untuk membuat *website* dinamis dan interaktif. Dinamis artinya, *website* tersebut bisa berubah-ubah tampilan dan kontennya sesuai kondisi tertentu, dan interaktinya dapat memberikan *feedback* bagi *user*".

Menurut Kadir (2018:236), "PHP adalah bahasa berbentuk skrip yang memungkinkan pembuatan aplikasi web yang dikendalikan oleh data".

Dari ketiga definisi di atas dapat disimpukan bahwa PHP adalah bahasa pemrograman script server side yang digunakan untuk membuat *website* dinamis dan interaktif yang dikendalikan oleh data.

2.2.4 *Framework* Laravel

Menurut Junirianto (2017:1), *Framework* adalah sebuah software untuk memudahkan para programmer untuk membuat sebuah aplikasi web yang di dalam nya ada berbagai fungsi diantaranya plugin, dan konsep untuk membentuk suatu sistem tertentu agar tersusun dan tersetruktur dengan rapih.

Menurut Yudhanto dan Prasetyo (2018:17), Laravel adalah salah satu *framework* PHP terbaik yang dikembangkan oleh *Taylor Otwell*, proyek laravel dimulai pada April 2011.

Laravel adalah pengembangan website berbasis MVP yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, dan untuk

meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu. (Prianto dan Bunyamin 2020: 62).

Menurut Yudhanto dan Prasetyo (2019:10), *Framework* adalah komponen programmer yang siap digunakan ulang kapan saja sehingga programmer tidak harus membuat script yang sama untuk tugas yang sama.

Menurut Abdulloh (2017:3), terdapat beberapa keunggulan dari *framework* Laravel yaitu sebagai berikut.

1. Laravel memiliki banyak fitur yang tidak dimiliki oleh *framework* lain.
2. Laravel merupakan *framework* PHP yang ekspresif, artinya sintaks pada Laravel menggunakan bahasa yang mudah dimengeri sehingga *programmer* pemula sekalipun akan mudah paham kegunaan suatu sintaks walaupun belum mempelajarinya.
3. Laravel memiliki dokumentasi yang cukup lengkap, bahkan setiap versinya memiliki dokumentasi tersendiri mulai dari cara instalasi hingga penggunaan fitur-fiturnya.
4. Laravel digunakan oleh banyak *programmer* sehingga banyak *library* yang mendukung Laravel yang diciptakan para *programmer* pecinta Laravel.
5. Laravel didukung oleh Composer sehingga *library-library* diperoleh dengan mudah dari internet menggunakan Composer.
6. Laravel memiliki *template engine* tersendiri yang diberi nama *blade* yang memudahkan kita menampilkan data pada *template* HTML.

Menurut Supardi dan Sulaeman (2019:2), ada beberapa fitur yang dimiliki oleh *framework* Laravel yaitu sebagai berikut:

a. ***Bundles***, yaitu sebuah fitur dengan sistem pengemasan modular dan tersedia beragam di aplikasi.
b. ***Eloquent ORM*** merupakan penerapan PHP lanjutan menyediakan *metode internal* dari pola "*active record*" yang mengatasi masalah pada hubungan objek *database*.
c. ***Application Logic*** merupakan bagian dari aplikasi menggunakan *controller* atau bagian *Route*.

d. ***Reverse Routing*** mendefinisikan relasi atau hubungan antara *Link* dan *Route*.
e. ***Restful controllers*** memisahkan logika dalam melayani *HTTP GET* dan *POST*.
f. ***Class Auto Loading*** menyediakan *loading* otomatis untuk *class* PHP.
g. ***View Composer*** merupakan kode unit logikal yang dapat dieksekusi ketika *view* sedang *loading*.
h. ***loC Container*** memungkinkan objek baru dihasilkan dengan pembalikan *controller*.
i. ***Migration*** merupakan penyedia sistem kontrol untuk skema *database*.
j. ***Unit Testing*** banyak tes untuk mendeteksi dan mencegah regresi.
k. ***Automatic Pagination***, menyederhanakan tugas dari penerapan halaman.

2.2.5 Laragon

Menurut Harianto, dkk. (2019:14), "Laragon adalah perangkat lunak yang mendukung banyak sistem operasi, berfungsi sebagai server diri sendiri/localhost laragon menyediakan services, tools, fitur mulai dari Apache, MySQL, PHP Server, Mamchaced, Redis, Composer, Xdebug, PhpMyAdmin, Cmder dan Laravel".

2.2.6 *Website*

Menurut Junirianto (2019:1), "*Website* adalah kumpulan informasi/ kumpulan page yang biasa diakses lewat jalur internet".

Sedangkan menurut Aziz, dkk (2019:2), "*Website* merupakan layanan atau alat tukar menukar data, informasi yang menggunakan konsep *client-server* dimana antara pengguna dan administrator dapat saling memberikan data atau informasi yang dapat memudahkan keduanya".

Menurut Sa'ad (2020:5), "*Website* merupakan kumpulan file yang terletak pada komputer yang terhubung ke internet".

Berdasarkan pengertian diatas *website* merupakan kumpulan informasi/kumpulan page yang memberikan data atau informasi yang terletak pada komputer dan terhubung ke internet.

2.2.7 *Sublime* Text

Menurut Rachmanto (2017:21), "*Sublime text* merupakan text editor kedua yang sering digunakan, selain untuk belajar sublime text juga digunakan untuk penulisan coding theme".

Menurut Supono dan Putratama (2018:14), "*Sublime text* merupakan perangkat lunak text editor yang digunakan untuk membuat atau mengedit suatu aplikasi. *Sublime text* mempunyai fitur *plugin* tambahan yang memudahkan programmer dan juga memiliki desain yang simple dan keren sehingga menjadi terkesan elegan untuk sebuah *syntax editor*".

Menurut Rerung (2018:7), *Sublime text* adalah aplikasi yang gratis maupun yang berbayar. Sublime text mempunyai banyak keunggulan seperti:

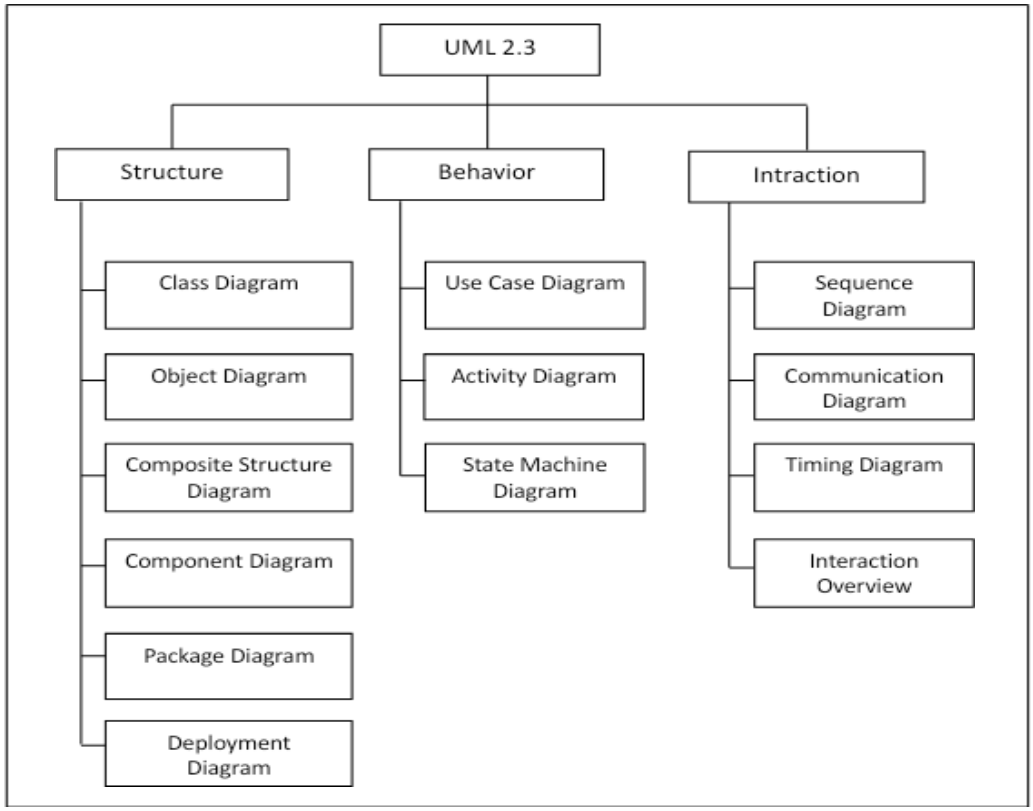
a. *Auto-Complation,*
b. *Minimap/Document Map,*
c. *Goto Defenition,*
d. *Split Editing,*
e. *Column Editing,*
f. *Multi Editing, dll.*

Jadi dapat disimpulkan, *Sublime text* adalah aplikasi editor kedua yang sering digunakan untuk kode dan teks yang gratis maupun yang berbayar yang dapat berjalan di berbagai *platform operating system* dengan menggunakan teknologi *phyton* API.

2.3 Teori Khusus

2.3.1 *Unified Modeling Language* (UML)

Menurut Sukamto dan Shalahuddin (2018:140), "Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori". Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar dibawah.



Gambar 2.1 Macam-macam Diagram UML

Penjelasan singkat dari pembagian kategori pada diagram UML menurut Sukamto dan Shalahuddin (2018:141):

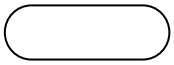
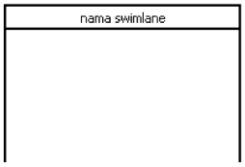
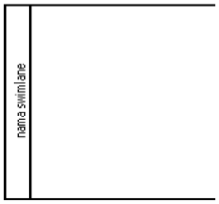
1) *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2) *Behavior diagrams* yaitu kumpulam diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahana yang terjadi pada sebuah sistem.
3) *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.3.1.1 *Activity Diagram*

Menurut Sukamto dan Shalahuddin (2018:161), menjelaskan tentang a*ctivity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Sukamto dan Shalahuddin (2018:162), Adapun simbol-simbol yang digunakan dalam *activity diagram* adalah sebagai berikut:

Tabel 2.1 Simbol-simbol pada *Activity Diagram*

No	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan/decision 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
4.	penggabungan/ *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
5.	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6.	Swimlane  Atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(Sumber: Sukamto dan Shalahuddin 2018:162)

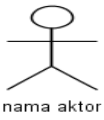
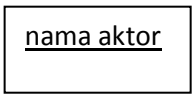
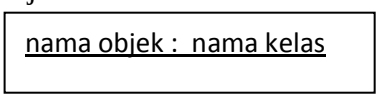
2.3.1.2 *Sequence Diagram*

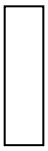
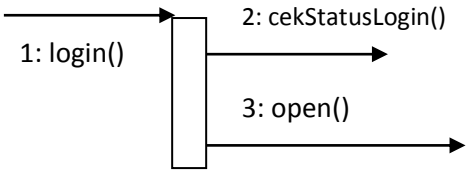
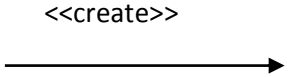
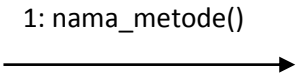
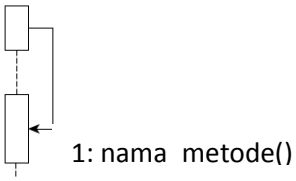
Sukamto dan Shalahuddin (2018:165), menjelaskan bahwa diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah beserta metode-metode yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram *sequence* juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

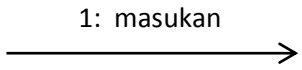
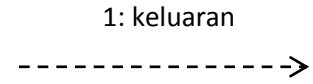
Banyaknya diagram *sequence* yang harus digambarkan adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.

Berikut simbol-simbol pada *Sequence Diagram*:

Tabel 2.2 Simbol-simbol pada *Sequence Diagram*

No	Simbol	Deskripsi
1.	Aktor  atau  tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakanorang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2.	Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek.
3.	Objek  	Menyatakan objek yang berinteraksi pesan.

No	Simbol	Deskripsi
4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  maka cekStatusLogin() dan open() dilakukan di dalam metode login() aktor tidak memiliki waktu aktif.
5.	Pesan tipe create 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat .
6.	Pesan tipe call 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.

No	Simbol	Deskripsi
7.	Pesan tipe send 	menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
8.	Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

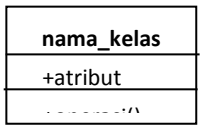
(Sumber: Sukamto dan Shalahuddin 2018:165-166)

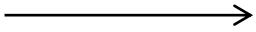
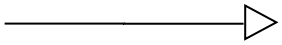
2.3.1.3 *Class Diagram*

Sukamto dan Shalahuddin (2018:141), menyebutkan *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Diagram Class* dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Sukamto dan Shalahuddin (2018:146), Adapun simbol-simbol yang digunakan dalam *class diagram* adalah sebagai berikut:

Tabel 2.3 Simbol-simbol pada *Class Diagram*

No	Simbol	Deskripsi
1.	kelas 	Kelas pada struktur sistem
2.	antarmuka / interface  **nama_interface**	Sama dengan konsep interface dalam pemrograman berorientasi objek

No	Simbol	Deskripsi
3.	asosiasi / association	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai *multiplicity*
4.	asosiasi berarah / *directed association* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
5.	Generalisasi 	Relasi antarkelas dengan makna generalisasi – spesialisasi (umum khusus)
6.	kebergantungan / *dependency*	Relasi antarkelas dengan makna kebergantungan antar kelas
7.	*Link*	Relasi antar *node*

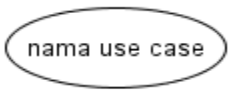
(Sumber: Sukamto dan Shalahuddin 2018:146-147)

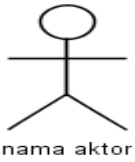
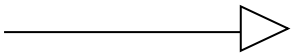
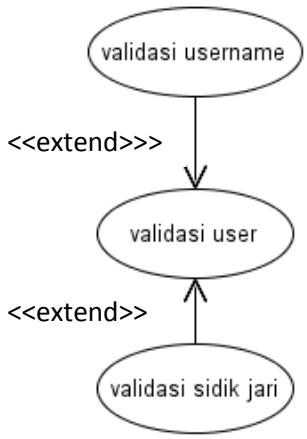
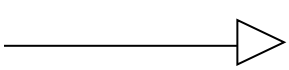
2.3.1.4 *Use Case Diagram*

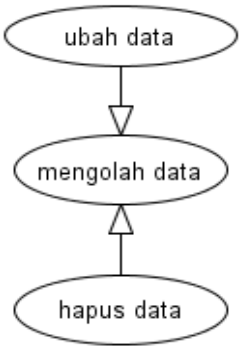
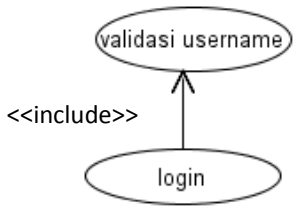
Menurut Sukamto dan Shalahuddin (2018:155) *"Use case* merupakan pemodelan untuk kelakuan *(behavior)* sisteminformasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat".

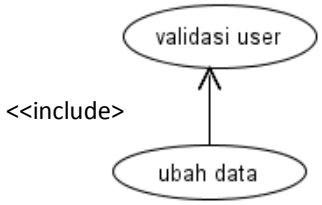
Menurut Sukamto dan Shalahuddin (2018:156) Adapun simbol-simbol yang digunakan dalam *use case* adalah sebagai berikut: Berikut simbol-simbol pada *Use Case Diagram*:

Tabel 2.4 Simbol-simbol pada *Use Case Diagram*

No	Simbol	Deskripsi
1.	*Use case* 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal-awal frase nama *use case*

No	Simbol	Deskripsi
2.	aktor / *actor*  	orang, proses, atau siatem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama actor
3.	asosiasi / *association* 	komunikasi antar aktor dan *use case* yang berpartisioasi pada *use case.*
4.	Generalisasi / *generalization* 	hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya
5.	Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:

No	Simbol	Deskripsi
		 arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
6.	Menggunakan / include / uses <<include>> <<uses>>	relasi tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini ada dua sudut pandang yang cukup besar mengenai include di *use case*: ● *Include* berarti *use case* yang ditambahkan akan selalu di panggil saat *use case* tambahan dijalankan, misalnya pada kasus berikut:  *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang di tambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:

No	Simbol	Deskripsi
		 kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

(Sumber: Sukamto dan Shalahuddin 2018:156-158)